<https://www.youtube.com/shorts/WDpc1olT2n0>

<https://www.youtube.com/shorts/nRfedP8BC8Y>

**https://www.youtube.com/shorts/oivxmsbRi9w**

############## Frequently Asked Interview question ################

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **emp.id** | **Name** | **dep.id** | **salary** | **manager\_id** |
|  |  |  |  |  |

* 1. How to find duplicate value in table

Ans: select emp.id,count(1) from Employee

group by emp.id

having count(1) > 1;

* 1. How to delete the duplicates from the table?

Ans. Here we can use CTE(common table expression) its like temporary table used to simplify the query .

To find duplicate values we will use Row\_num() function.

Here 1st write row number query >>>> CTE>>>> Upper query:

Select \* from Employee

With CTE AS

(Select \*,row\_number() over(partition by emp\_id order by emp.id) as Rw\_Num

From Employee)

Delete from CTE where Rw\_Num > 1;

* 1. Difference between Union & Union All

Ans. Union All will give all values uncluding duplicates Whereas Union will give only unique values.

* 1. Difference between Rank,Dense rank & Row number:

Ans:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp.id** | **Name** | **dep.id** | **salary** |
| 1 | Ank | 100 | 10000 |
| 2 | Moh | 100 | 15000 |
| 3 | Raj | 100 | 10000 |
| 4 | Tej | 100 | 5000 |
| 5 | Ketan | 200 | 12000 |
| 6 | Sneha | 200 | 12000 |
| 7 | siddhi | 200 | 9000 |
| 8 | Keshar | 200 | 5000 |

Select \*,

Rank() OVER (order by salary desc) as Rank,

Dense\_Rank() OVER(order by salary desc ) as Dense\_Rank,

Row\_Number() OVER(order by salary desc ) as Row\_Number

FROM Employee e;

So result is ranked overall by salary desc:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **emp.id** | **Name** | **dep.id** | **salary** | **Rank** | **Dense\_Rank** | **Row\_Number** |
| 1 | Ank | 100 | 15000 | **1** | 1 | 1 |
| 2 | Moh | 200 | 12000 | 2 | 1 | 2 |
| 3 | Raj | 200 | 12000 | 2 | 1 | 3 |
| 4 | Tej | 100 | 10000 | 4 | 3 | 4 |
| 5 | Ketan | 100 | 10000 | 4 | 3 | 5 |
| 6 | Sneha | 200 | 9000 | 6 | 4 | 6 |
| 7 | siddhi | 200 | 5000 | 7 | 5 | 7 |
| 8 | Keshar | 100 | 5000 | 7 | 5 | 8 |

# If we want Ranks basis on each department salary:

Select \*,

RANK() OVER (PARTITION BY Dep\_Id ORDER BY salary desc) as Rank,

DENSE\_RANK() OVER(PARTITION BY Dep\_Id ORDER BY salary desc ) as Dense\_Rank,

ROW\_NUMBER() OVER(PARTITION BY Dep\_Id ORDER BY salary desc ) as Row\_Number

FROM Employee e;

Result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **emp.id** | **Name** | **dep.id** | **salary** | **Rank** | **Dense\_Rank** | **Row\_Number** |
| 1 | Ank | 100 | 15000 | **1** | 1 | 1 |
| 2 | Moh | 100 | 10000 | 2 | 2 | 2 |
| 3 | Raj | 100 | 10000 | 2 | 2 | 3 |
| 4 | Tej | 100 | 5000 | 4 | 3 | 4 |
| 5 | Ketan | 200 | 12000 | 1 | 1 | 5 |
| 6 | Sneha | 200 | 12000 | 1 | 1 | 6 |
| 7 | siddhi | 200 | 9000 | 3 | 2 | 7 |
| 8 | Keshar | 200 | 5000 | 4 | 3 | 8 |

**Q. 5. Find department wise highest salary :**

We will use Rank function:

Select \* from

(Select \*,

Rank() OVER(PARTITION BY Dep\_Id ORDER BY Salary desc) AS Rnk

FROM Employee) AS A

Where A.Rnk = 1;

1. **Find the employee which are not in dep. Table:**

Employee:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp.id** | **Name** | **dep.id** | **salary** |
| 1 | Ank | 100 | 10000 |
| 2 | Moh | 100 | 15000 |
| 3 | Raj | 100 | 10000 |
| 4 | Tej | 100 | 5000 |
| 5 | Ketan | 200 | 12000 |
| 6 | Sneha | 200 | 12000 |
| 7 | siddhi | 200 | 9000 |
| 8 | Keshar | 200 | 5000 |

Department Table:

|  |  |
| --- | --- |
| **Dep.id** | **Dep\_Names** |
| 100 | Analytics |
| 300 | IT |

1. Select \* from Employee where Dep.id NOT IN In (select Dep.id from Department) ;
2. Select E.\*,D.\* from Employee E

Left Join Dapartment D

ON E.Dep\_Id=D.Dep\_Id where E.Dep\_Id = D.Dep\_id

Where Dep\_Name is NULL;

* 1. Find 2nd Highest salary in each department;

Ans: select \* from

(select Dense\_Rank() OVER (Partition by Dep\_Id ORDER BY Salary desc) AS D\_Rank From Employee

) A

where A.D\_Rank= 2;

* 1. Find all Transactions done by ‘SHILPA’

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer\_Name** | **Order date** | **Order amount** | **Gender** |
| SHILPA | 05-08-2024 | 2 | F |
| Raj | 12-08-2024 | 4 | M |
| Mohan | 05-07-2024 | 4 | M |
| shilpa | 05-06-2024 | 1 | F |

Ans: Select \* from Order where Upper(Customer\_Name) = ‘SHILPA’

* 1. Update query to swap gender

Ans: We cant use simply Update statement where Gender = Male make it female likewise. So use below way to tackle this issue:

Update Order SET Gender =

Case When Gender = ‘F’ Then ‘M’

When Gender = ‘M’ Then ‘F’

End;

* 1. What is order of Execution in SQL?

Ans: From > Where > Group BY > Having > Select > Order BY > Limit

* 1. Find Monthly sales & sort it by Desc order.

|  |  |
| --- | --- |
| **Order\_Date** | **Sales** |
| 2024-01-01 | 10 |
| 2024-01-15 | 15 |
| 2024-02-03 | 20 |
| 2024-02-04 | 30 |
| 2024-05-05 | 40 |
| 2024-03-05 | 50 |
| 2024-03-10 | 60 |

Ans:

Select Extract(Year from Order\_Date) as Year,

to\_char(Order\_date,’Mon’) AS Months,

Sum(Sales) as Total\_Sales

From Products

Group by 1,2

Order by 3;

Output:

|  |  |  |
| --- | --- | --- |
| **Year** | **Months** | **Total\_Sales** |
| 2024 | 1 | 25 |
| 2024 | 2 | 50 |
| 2024 | 3 | 110 |
| 2024 | 5 | 40 |

* 1. Find the candidate suitable for Data science job having skills: Python,SQL,PowerBi.

Write a query to list the candidates possess required skills as above & sort it by Candidate id asc.:

|  |  |
| --- | --- |
| **Cand\_Id** | **Skills** |
| 101 | Power BI |
| 101 | Python |
| 101 | SQL |
| 102 | SQL |
| 108 | Python |
| 108 | SQL |
| 108 | Power BI |
| 104 | SQL |

Ans: Select Cand\_Id, Count(skills)

from DS

Where Skills IN (‘Python’,’SQL’,’PowerBI’)

Group by Cand\_Id

Having Count(Skills) = 3

Order by Cand\_Id asc;

* 1. You have table Named A with input is n= 1,2,3,4,5 as input. Write query to get output as
     1. 1 times , 2- 2 times ,3- 3 times like wise upto 5

Ans:

Create TABLE Numbers (n int);

Insert into Numbers

Values (1),(2),(3),(4),(5);

Solution to Que: 1) With CTE AS

(

Select n, 1 as num\_counter from numbers

UNION ALL

Select n, num\_counter+1 as num\_counter from CTE

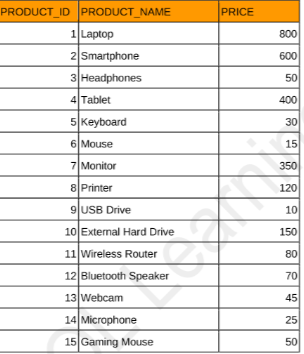
Where num\_counter+1 <= n

)

Select n from CTE

Order by n;

* 1. You are provided with a table named Products containing information about various products, including their names and prices. Write a SQL query to count number of products in each category based on its price into three categories below. Display the output in descending order of no of products.  
        
     1- "Low Price" for products with a price less than 100  
     2- "Medium Price" for products with a price between 100 and 500 (inclusive)  
     3- "High Price" for products with a price greater than 500.  
     Tables: Products



Ans: Using CTE as:

With CTE AS (

Select \* ,Case

WHEN price < 100 THEN 'Low Price'  
 WHEN price between 100 AND 500 THEN 'Medium Price'  
 ELSE 'High Price'

END As Category

FROM Products

)

Select Category,Count(\*) as Num\_of\_Products from CTE

Group by category

Order by Num\_of\_Products;

**Que . 1 We have table data User with columns: user id | Transaction\_Date | Trans\_Id .**

**How to check May customers,June customers ,Ratio using SQL?**

WITH May\_Customers AS (

SELECT DISTINCT user\_id

FROM User

WHERE EXTRACT(MONTH FROM Transaction\_Date) = 5

AND EXTRACT(YEAR FROM Transaction\_Date) = 2024 -- Adjust the year as needed

),

June\_Customers AS (

SELECT DISTINCT user\_id

FROM User

WHERE EXTRACT(MONTH FROM Transaction\_Date) = 6

AND EXTRACT(YEAR FROM Transaction\_Date) = 2024 -- Adjust the year as needed

)

SELECT COUNT(DISTINCT May\_Customers.user\_id) AS May\_Customers\_Count,

COUNT(DISTINCT June\_Customers.user\_id) AS June\_Customers\_Count,

(COUNT(DISTINCT June\_Customers.user\_id) \* 1.0 / COUNT(DISTINCT May\_Customers.user\_id)) AS Customer\_Ratio

FROM May\_Customers

LEFT JOIN June\_Customers

ON May\_Customers.user\_id = June\_Customers.user\_id;

**OR**

WITH May AS (

SELECT COUNT(DISTINCT user\_id) AS May\_Customers

FROM User

WHERE EXTRACT(MONTH FROM Transaction\_Date) = 5

AND EXTRACT(YEAR FROM Transaction\_Date) = 2024

), June AS (

SELECT COUNT(DISTINCT user\_id) AS June\_Customers

FROM User

WHERE EXTRACT(MONTH FROM Transaction\_Date) = 6

AND EXTRACT(YEAR FROM Transaction\_Date) = 2024

)

SELECT June.June\_Customers,

May.May\_Customers,

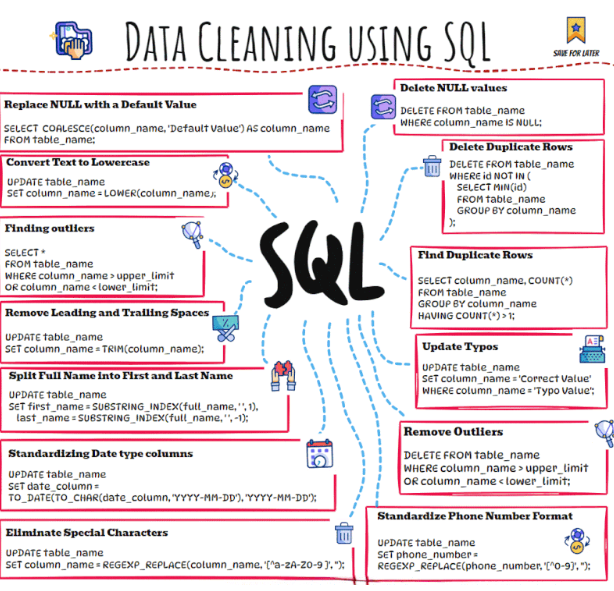
(June.June\_Customers \* 1.0 / May.May\_Customers) AS Customer\_Ratio

FROM June, May;

Que. What is data analytics?

Ans: Data Analytics is the process of examining and interpreting data to extract meaningful insights.

* It involves collecting, cleaning, and transforming data
* Applying statistical and machine learning techniques to analyze data
* Visualizing and communicating insights to stakeholders
* Examples include predicting customer behavior, optimizing business processes, and identifying trends in healthcare data



* 1. Scenario based questions:

how to analyze the root cause analysis to check the reduced transactions in PhonePay?

Ans: We will consider below factors:

**Data Collection:**

ather relevant data to analyze the potential causes of the problem.

* Transaction Data: Historical data of transactions, including trends before and after the decline.
* Customer Feedback: Surveys, support tickets, app store reviews, and social media posts that provide insights into customer pain points.
* Technical Metrics: App performance metrics, including transaction success rate, app crashes, latency issues, and downtime.
* Competitor Analysis: Compare your app’s features, pricing, and promotions with competitors like Google Pay, Paytm, etc.
* External Factors: Consider economic downturns, regulatory changes, and shifts in market behavior.

**Identify Potential Causes:**

1) User Experience Issues:

* App Crashes/Performance: Slow or buggy app experience can deter users from completing transactions.
* Interface Changes: Recent UI/UX changes might have confused or alienated users.
* Complex Navigation: Increased steps in transaction completion might discourage users.

2) Technical Glitches:

* Downtime or Service Outages: If the platform experienced outages, users may have switched to competitors.
* Transaction Failures: Frequent payment failures due to server or network issues could reduce trust.

3) External Competition:

* Competitor Promotions: Competitors might be offering cashback or discounts that pull users away.
* User Shifts: Users may shift to alternative payment options like Google Pay or UPI.

4) Marketing and Promotions:

* Reduction in Offers: If previous offers or discounts expired, users may have decreased their transaction frequency.
* Ad Spend: Less advertising or poor targeting may have led to reduced customer acquisition.

5) Demographic or Behavioral Shifts:

* Economic Factors: Unemployment, inflation, or other economic issues could affect user spending.
* User Segments: Certain age groups or regions may have shown a higher drop in transactions.

**Step 4: Analyze the Data**

Use both qualitative and quantitative methods to validate the causes:

* **Trend Analysis:** Compare transaction volumes across different times, dates, regions, and user segments to identify patterns.
* **A/B Testing:** If changes were made to the app or service, analyze how different versions affected user behavior.
* **Correlation Analysis:** Identify whether factors like app performance, offers, or user feedback correlate with the decline in transactions.
* **Customer Segmentation:** Break down the data to see if certain user groups (age, location, frequency) have reduced activity more than others.
  1. Problem: You are given two tables; employees and projects.  
       
     Table 1: employees  
     e\_id   e\_name    department  
     1    John Doe    Finance  
     2    Jane Smith    IT  
     3    Mike Brown    Finance  
     4    Emma Wilson    IT  
     5    Sarah Davis    Finance
  2. Table 2: projects  
     p\_id   e\_id    p\_name    p\_hours  
     101  1    Project A    100  
     102    2   Project B    150  
     103    3    Project A    120  
     104    4    Project C    130  
     105    5    Project B    140  
       
     **Write an SQL query to find the top 2 employees from the Finance department who have worked the most hours across all projects**.

Solution:

select E.e\_id ,sum(p\_hours) as Total\_hours from Employee E

Inner join projects P on E.eid = P.e\_id

Where department = ‘Finance’

Group by 2

Order by 2 desc

* 1. employee\_id is the primary key for this table.  
     Each row of this table contains the ID of each employee and their respective team.  
     **Write an SQL query to find the team size of each of the employees.**  
       
     Return result table in any order.  
       
     SQL Script:  
     Table: Employee  
       
     +---------------+---------+  
     | Column Name | Type |  
     +---------------+---------+  
     | employee\_id | int |  
     | team\_id | int |  
     +---------------+---------+

Solution:

Select E.emp\_id,A.Team\_Size

From Employee E

Join

(Select team\_id, count(emp\_id) as Team\_Size

From Employee

Group by team\_id) A

ON E.team\_id = A.team\_id

OR :

Select emp\_id ,

Count(emp\_id) over (partition by team\_id) as team\_size

From Employee

Order by emp\_id asc;

(here in above query

**COUNT(emp\_id) OVER (PARTITION BY team\_id)**:

* This calculates the total number of employees (COUNT(emp\_id)) for each team (team\_id), which represents the size of that team.
* The PARTITION BY clause ensures that the count is reset for each different team\_id, so it only counts employees within the same team.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | |  |  |  |
|  | | | |  |  |  |
| **####################### EXL Interview questions:**  1.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | SaleID | ProductID | SaleDate | QuantitySold | salesAmount | | 1 | 1 | 01-01-2023 | 10 | 100 | | 2 | 2 | 01-01-2023 | 5 | 50 | | 3 | 3 | 01-01-2023 | 15 | 20 | | 4 | 1 | 02-01-2023 | 15 | 150 | | 5 | 2 | 02-01-2023 | 7 | 70 | | 6 | 1 | 03-01-2023 | 12 | 120 | | 7 | 2 | 03-01-2023 | 6 | 60 | | 8 | 3 | 03-01-2023 | 16 | 30 | | Find current day sales & Previous day sales in sql with output as below:   |  |  |  | | --- | --- | --- | | ProductId | SaleDate | DifferenceFromPrevious | | 1 | 01-01-2023 |  | | 1 | 02-01-2023 | 50 | | 1 | 03-01-2023 | -30 | | 2 | 01-01-2023 |  | | 2 | 02-01-2023 | 20 | | 2 | 03-01-2023 | -10 | | 3 | 01-01-2023 |  | | 3 | 03-01-2023 | 10 | | | | | | |  |  |  |  |  |  |
|  | |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  | | | | | | |
| Sol:  WITH SalesWithPrevious AS  (  SELECT  s.ProductID,  s.SaleDate,  SUM(s.QuantitySold) AS QuantitySold,  SUM(s.salesAmount) AS salesAmount,  LAG(SUM(s.salesAmount)) OVER (PARTITION BY s.ProductID ORDER BY s.SaleDate) AS previous\_salesAmount  FROM SalesData s  GROUP BY s.ProductID, s.SaleDate  )  SELECT ProductID, SaleDate,  (salesAmount - previous\_salesAmount) AS difference\_from\_previous  FROM SalesWithPrevious  WHERE previous\_salesAmount IS NOT NULL  ORDER BY SaleDate; |  |  |  |  |  |  |

**########### ICICI Interview Questions ############**

Table A Table B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Col1 Name | Col2 Id |  |  | Col3 Name | Col4 Preference |
| a | 1 |  |  | a | apple |
| b | 2 |  |  | a | orange |
| b | 3 |  |  | b | grapes |
| c | 4 |  |  | d | banana |
| c | 5 |  |  | null | litchi |
| c | 6 |  |  | null | guava |
| d | 7 |  |  |  |  |
| null | 8 |  |  |  |  |

q. What is the output of Left join of Table A left join Table B ?

ans:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name\_A** | **Id** | **Name\_B** | **Preference** |
| a | 1 | a | apple |
| a | 1 | a | orange |
| b | 2 | b | grapes |
| b | 3 | b | grapes |
| c | 4 | NULL | NULL |
| c | 5 | NULL | NULL |
| c | 6 | NULL | NULL |
| d | 7 | d | banana |
| NULL | 8 | NULL | litchi |
| NULL | 8 | NULL | guava |

############################ SQL Practice questions for Interviews ########################################

* + 1. **Explain the difference between INNER JOIN and OUTER JOIN?**

Ans: An **INNER JOIN** returns only the rows where there is a match in both tables.

An **OUTER JOIN** returns all the rows from one (or both) of the tables, even if there is no match. If no match is found, NULL values are returned for columns from the table that doesn't have a corresponding match.

* + 1. **Write a query to fetch the second-highest salary from an employee table?**

Ans:

With SalaryRank AS

( Select \*, ROW\_NUMBER() OVER (ORDER BY salary DESC) AS rank

From employees )

Select salary AS 2nd\_highest\_salary

From SalaryRank

Where rank = 2;

* + 1. **How would you find duplicate rows in a table and delete them?**

**Ans:**

Finding Duplicate Rows:

SELECT name, email, phone\_number, COUNT(\*) AS duplicate\_count

FROM employees

GROUP BY name, email, phone\_number

HAVING COUNT(\*) > 1;

Deleting Duplicate Rows:

WITH CTE AS

( SELECT \*, ROW\_NUMBER() OVER (PARTITION BY name, email, phone\_number ORDER BY id) AS row\_num

FROM employees )

DELETE FROM CTE

WHERE row\_num > 1;

* + 1. **Describe how you would use GROUP BY and HAVING to filter grouped data.**

Ans: In SQL, the GROUP BY and HAVING clauses are used together to group rows based on one or more columns and then filter the resulting groups.

The **GROUP BY** clause groups rows that have the same values in specified columns into summary rows, like "total count" or "average."

It's often used with aggregate functions such as COUNT(), SUM(), AVG(), MAX(), or MIN() to compute results for each group.

e.g:

SELECT customer\_id, SUM(total\_amount)

AS total\_spent FROM orders

GROUP BY customer\_id;

The **HAVING** clause is used to filter groups after the GROUP BY operation.

It works like the WHERE clause, but while WHERE filters rows before they are grouped, HAVING filters the results of the GROUP BY operation after aggregation.

E.g If you want to find customers who have spent more than $100 in total:

SELECT customer\_id, SUM(total\_amount) AS total\_spent FROM orders GROUP BY customer\_id HAVING SUM(total\_amount) > 100;

**# Key differences:**

 **GROUP BY** aggregates the data.

 **HAVING** filters the aggregated data after the grouping process.

* + 1. **How do you handle NULL values in SQL, and why is this important?**

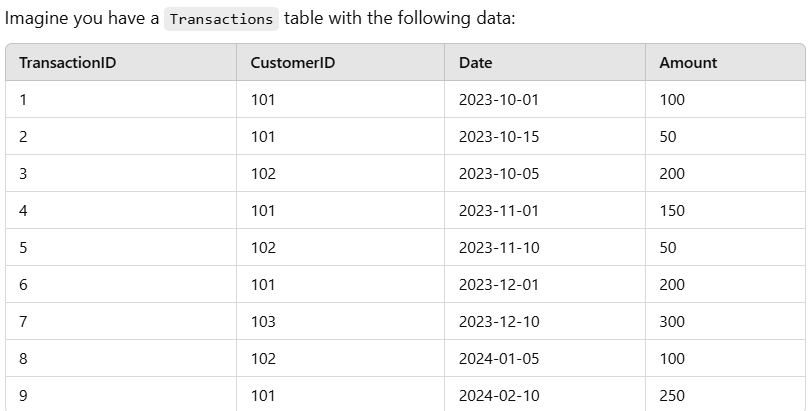
**Ans:**

* Use IS NULL or IS NOT NULL to check for NULL values in SQL.
* Use **COALESCE()**, **NULLIF()**, or database-specific functions like **IFNULL()** to handle or replace NULL values.
* Proper handling of NULL is important for **accurate reporting**, **data integrity**, and ensuring your queries perform as expected.

**################## Shelll Data analyst ###################################**

* + 1. **You have a table Transactions with columns TransactionID, CustomerID, Date, and Amount. Write a query to calculate the cumulative revenue per customer for each month in the last year.**

**Sol:**



Select

CustomerID,

Date Format(Date,’%Y-%m’) as Month, -- Format the date as Year-Month

SUM(Amount) as Monthly Revenue, -- Monthly revenue per customer

SUM(Sum(Amount)) OVER(Partition by Customer Order BY Rows between UNBOUNDED Preceeding AND Current Row) AS Cumulatice\_Revenue

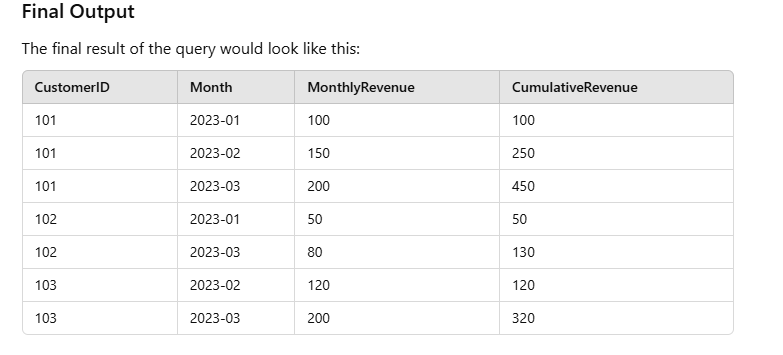
-- Cumulative revenue

FROM Transaction

Where Date >= Date\_Sub(Curdate(), Interval 1 Year) -- Filter for the last year

Group by 1,2 -- Group by Customer and Month

Order by 1,2;



**Que. 7 Using the Employee table (EmpID, ManagerID, JoinDate, Dept, Salary):  
 • Find the nth highest salary.  
 • Identify all employees under a specific manager, including their subordinates at any level.  
 • Find the cumulative salary department-wise for employees who joined in the last 30 days.**

Sol:

* 1. **Find the nth highest salary.**

With CTE AS (

Select Salary, Denserank() OVER(Order by salary desc) as Drnk

FROM Employee)

Select Salary from CTE where Drnk = <N> ;

* 1. **Find the cumulative salary department-wise for employees who joined in the last 30 days**

Select Dept,

SUM(salary) as Cumulative\_Salary

FROM Employee

Where Join\_Date >= Now() – Interval 30 days

Group by Dept;

OR

`

Select Dept,

SUM(salary) as Cumulative\_Salary

FROM Employee

Where DATEDIFF(DAY, Join\_Date, GETDATE()) Between 1 And 30

Group by Dept

Order by Join\_Date desc;

**Que. 8 Get the second most recent activity, if there is only one activity then return that one.**

create table UserActivity(username varchar(20),activity varchar(20),startDate Date,endDate Date);

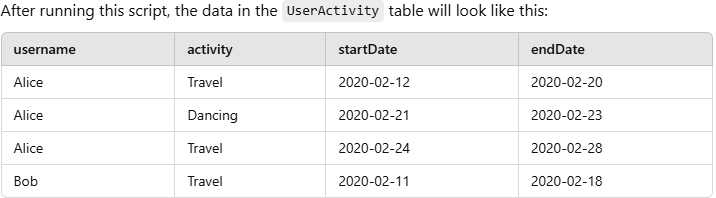
insert into UserActivity

values ('Alice','Travel','2020-02-12','2020-02-20') ,

('Alice','Dancing','2020-02-21','2020-02-23'),

('Alice','Travel','2020-02-24','2020-02-28'),

('Bob','Travel','2020-02-11','2020-02-18');



Sol:

With CTE AS (

Select Username,Activity ,

count(\*) over (partition by username) as count, ----- Counts total activities per user

Rank() OVER(Partition BY Username Order by StartDate asc) as Rnk ------ Assigns rank based on start date

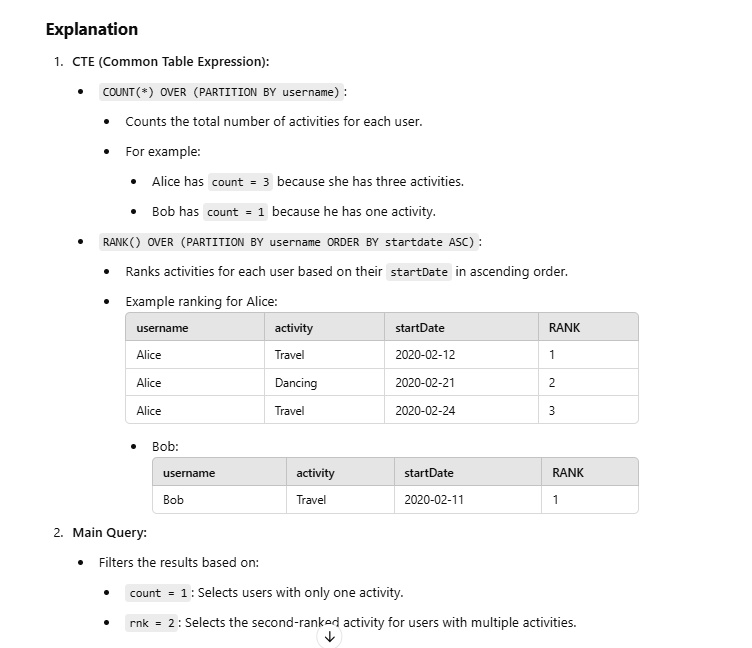
FROM Useractivity

)

Select Username , Activity from CTE

Where Count = 1 OR Rnk = 2 ------ Selects users with only one activity or the second-ranked activity





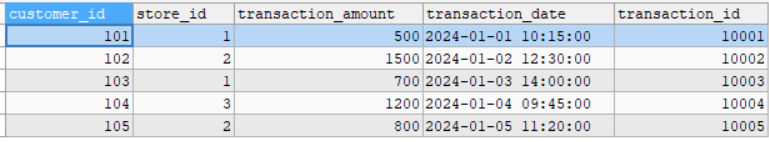
**# Visa Data Analyst Interview Questions:**

# Q. 9 Identify the top 3 areas with the highest customer density.

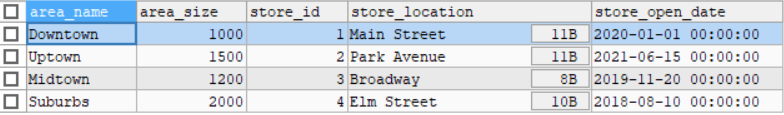
-- Customer density = (total number of unique customers in the area / area size).

-- Your output should include the area name and its calculated customer density.

SELECT \* FROM transaction\_records;



SELECT \* FROM stores;



# Sol:

SELECT area\_name ,

COUNT(DISTINCT customer\_id )/ area\_size AS Cust\_Density

FROM stores s

JOIN transaction\_records t ON s.store\_id = t.store\_id

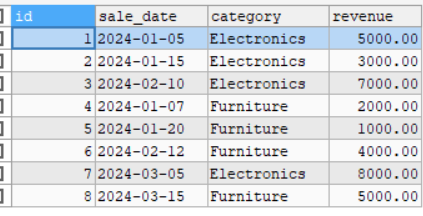
GROUP BY area\_name ,area\_size

ORDER BY 2 DESC Limit 3;

Que. 10 PWC Interview Questions

**a) Write a query to find the cumulative revenue by month for each product category in a sales table.**

Sale\_Date | Revenue | Category



Sol:

WITH Monthly\_Revenue AS (

SELECT category,

DATE\_FORMAT(sale\_date, '%Y-%m' ) AS MONTH,

SUM(revenue) AS Monthly\_Rev

FROM sales\_pwc

GROUP BY 1,2) ,

Cumulative\_Revenue AS(

SELECT category,

MONTH,

Monthly\_Rev,

SUM(Monthly\_Rev) OVER(PARTITION BY category ORDER BY MONTH) AS Cumulative\_Sum

FROM Monthly\_Revenue

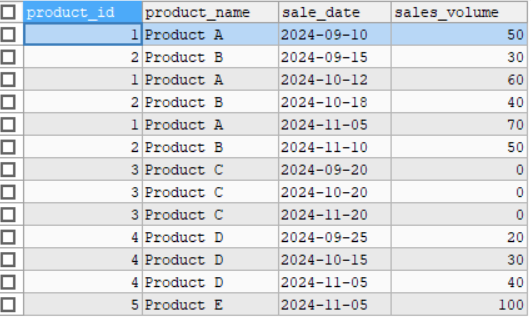
)

SELECT MONTH , category ,Monthly\_Rev ,Cumulative\_Sum

FROM Cumulative\_Revenue

ORDER BY category, MONTH;

**Que. b) How would you retrieve the top 5 products BY sales volume, excluding ANY products that had zero sales IN the past 3 months?**



**Sol:**

WITH RecentSales AS (

SELECT

product\_id,

product\_name,

SUM(sales\_volume) AS Total\_sal\_vol

FROM sales\_prod

WHERE sale\_date >= DATE\_SUB(CURDATE() ,INTERVAL 3 MONTH )

GROUP BY 1,2

HAVING SUM(sales\_volume) > 0

),

RankedProducts AS (

SELECT

product\_id,

product\_name,

Total\_sal\_vol,

RANK() OVER(ORDER BY Total\_sal\_vol DESC) AS Rnk

FROM RecentSales

)

SELECT product\_id,

product\_name,

Total\_sal\_vol FROM RankedProducts

WHERE Rnk <= 5;